# 姚攀的博客 1.01^365=31.78

Engineers are versatile minds who create links between science, technology, and society.



## (spring全家桶十)Spring Statemachine有限状态机与地址分析

标签： 有限状态机 state machine spring

2017-09-23 15:47 897人阅读 评论(0)

分类：

J2EE（34）

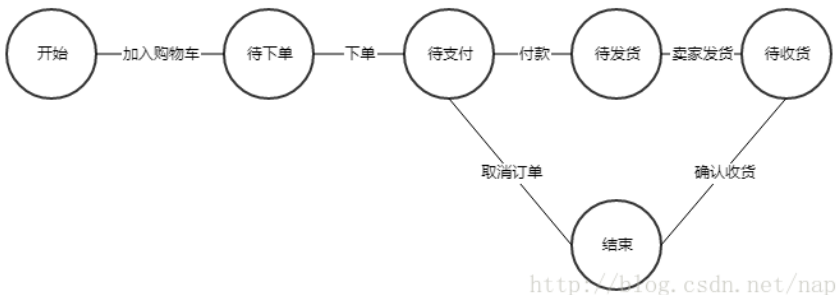


目录(?) [+]

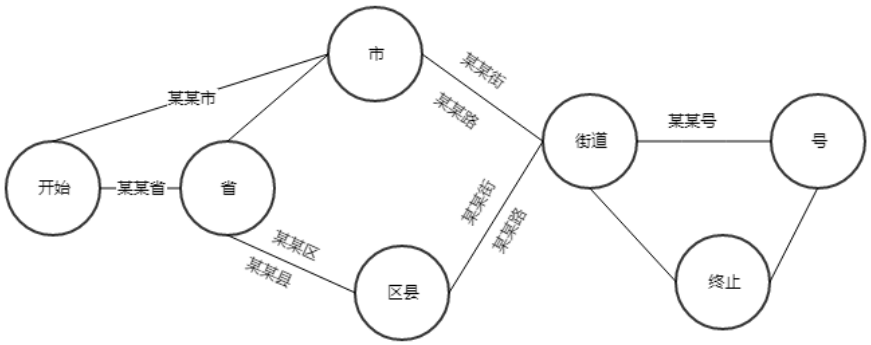
# 一、有限状态机

有限状态机是一个特殊的有向图，包含节点和连接这些节点的弧。每个有限状态机都有开始、结束和若干个中间状态，每个弧上带有从一个状态进入下一个状态的条件。

以一个简化的购物流程为例，开始和结束之间有待下单、待支付、待发货、待收货四个状态，从一个状态转向另外一个状态中间需要发送事件。




有限状态机可以用于中文地址分析，识别地址的有限状态机如下。




识别地址的有限状态机 http://blog.csdn.net/

给出一个地址，如果当前状态是“省”，后面一个词组是二级市，就进入状态“市 个词组是某某区或者某某县，就进入“区县”状态。
如果一个地址能从状态机的开始状态经过若干中间状态到达终止状态，地址有效。 无效。

例如：北京市海淀区清华东路17号，有效。河北省大连市友好路，无效。

# 二、Spring Statemachine

Spring Statemachine是spring的有限状态机开源框架，诞生不久。

Spring Statemachine框架的目标：
1.简单易用
2.采用层次化状态机结构简化复杂状态配置
3.State machine regions提供复杂的状态机配置
4.使用triggers, transitions, guards and actions.
5.类型安全的适配器配置
6.用于Spring应用程序上下文之外的简单实例化的生成器模式
7.基于ZooKeeper实现的分布式状态机
8.有事件监听器
9.在持久层存储状态配置
10.Spring IOC集成，bean可以和状态机交互

# 三、实例

## 3.1 创建spring boot工程

访问：http://start.spring.io/ ，新建一个spring boot工程：

文章：13篇

阅读：17571

文章分类

Elasticsearch (43)
Lucene (15)
hadoop (16)
J2EE (35)
数据结构 (16)
前端 (16)
ruby (18)
机器学习 (4)
python (5)
linux (11)
latex (2)
数据库 (12)
spring (9)
tools (1)
hibernate (1)
mybatis (1)
生活 (1)

文章存档

2017年12月 (9)
2017年11月 (1)
2017年10月 (3)
2017年09月 (2)
2017年08月 (1)

展开

最新评论

MapReduce编程(六) 从HDFS导入数据到El…
qq_38094271 ：用最新版本的那个 成功了。注意json 文件最后一行要换行

《从Lucene到Elasticsearch:全文检索实战…
牙小木 ：书已买，正在看

Lucene扩展停用词字典与自定义词库
yyyseb ：@cuipeng6561:这个文件在ik的目录下。

Lucene扩展停用词字典与自定义词库
cuipeng6561 ：IKAnalyzer.cfg.xml 这个文件是要自己写么?

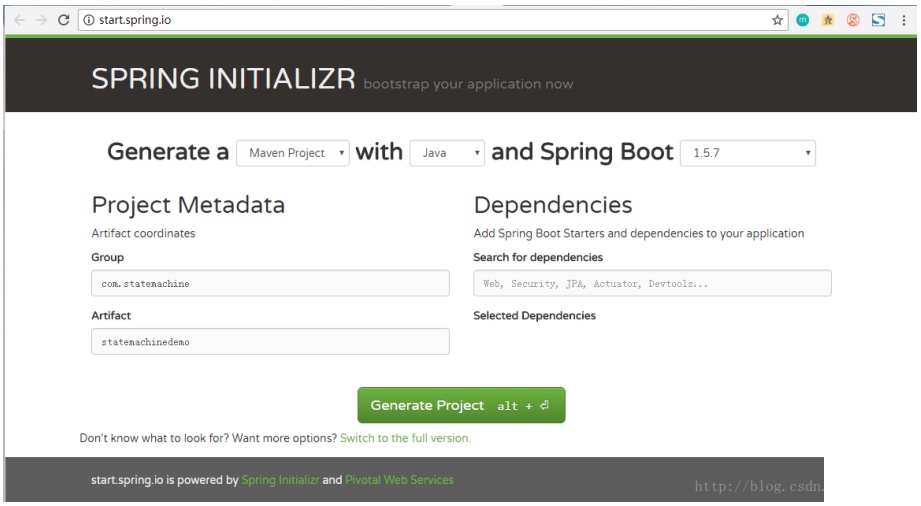
新人千万不要在 Windows 上使用 Ruby on …
韩大喵 ：对于新手的我来说的，相当中肯！

《从Lucene到Elasticsearch:全文检索实战…
艾鹤 ：6666，恭喜发财。

Elasticsearch 5.X下JAVA API使用指南
qq_41296216 ：@napoay:maven坐标是 <dependency> ...

Elasticsearch java api(五) Bulk批量索引
yyyseb ：@wangmaohong0717:可以不指定，会自动生成。

Elasticsearch 5.X下JAVA API使用指南
yyyseb ：@qq_41296216:贴一下maven坐标吧，或者，博客上有QQ群，加群讨论




pom.xml：

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.o
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.statemachine</groupId>
    <artifactId>statemachinedemo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>statemachinedemo</name>
    <description>Demo project for Spring Boot</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.5.7.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncodi
        <java.version>1.8</java.version>
    </properties>

    <dependencies>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
```

```
        </dependency>

        <dependency>
            <groupId>org.springframework.statemachine</groupId>
            <artifactId>spring-statemachine-core</artifactId>
            <version>1.2.6.RELEASE</version>
        </dependency>

        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.17</version>
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>


</project>
```

创建状态：

```
public enum States {
    START,     //开始
    PROVINCE,  //省
    CITY,      //市
    DISTRIC,    //区县
    STREET,     //街道
    STREET_NUM,  //街道号
}
```

定义事件：

```
public enum Events {
    HAS_PROVINCE,    //地址中有一级省份
    HAS_CITY,        //地址中有二级市
    HAS_DISTRIC,     //地址中有三级县或者区
    HAS_STREET,      //地址中有四级的街道
    HAS_STREET_NUM   //地址中有五级的街道号
}
```

设置状态之间的转换关系：

```
import com.sun.org.apache.regexp.internal.RE;
import org.apache.log4j.Logger;
```

```
3  import org.springframework.context.annotation.Configuration;
4  import org.springframework.statemachine.config.EnableStateMachine;
5  import org.springframework.statemachine.config.EnumStateMachineConfigurerAdapte
6  import org.springframework.statemachine.config.builders.StateMachineConfigurati
7  import org.springframework.statemachine.config.builders.StateMachineStateConfig
8  import org.springframework.statemachine.config.builders.StateMachineTransitionC
9  import org.springframework.statemachine.listener.StateMachineListener;
10 import org.springframework.statemachine.listener.StateMachineListenerAdapter;
11 import org.springframework.statemachine.transition.Transition;
12
13 import java.util.EnumSet;
14
15
16 @Configuration
17 @EnableStateMachine
18 public class Config extends EnumStateMachineConfigurerAdapter<States, Events> {
19     public static final Logger logger = Logger.getLogger(Config.c
20
21     @Override
22     public void configure(StateMachineStateConfigurer<States, Events> states) t
23         states
24                 .withStates()
25                 .initial(States.START)
26                 .states(EnumSet.allOf(States.class));
27     }
28
29     @Override
30     public void configure(StateMachineTransitionConfigurer<States, Events> tran
31         transitions
32                 .withExternal().source(States.START).target(States.PROVINCE).ev
33                 .and()
34                 .withExternal().source(States.START).target(States.CITY).event(
35                 .and()
36                 .withExternal().source(States.PROVINCE).target(States.CITY).eve
37                 .and()
38                 .withExternal().source(States.PROVINCE).target(States.DISTRIC).
39                 .and()
40                 .withExternal().source(States.CITY).target(States.STREET).event
41                 .and()
42                 .withExternal().source(States.STREET).target(States.STREET_NUM)
43     }
44
45     public StateMachineListener<States, Events> listener() {
46         return new StateMachineListenerAdapter<States, Events>() {
47             @Override
48             public void transition(Transition<States, Events> transition) {
49                 if (transition.getTarget().getId() == States.START) {
50                     logger.info("开始:");
51                     return;
52                 }
53
54                 if (transition.getSource().getId() == States.START &&
55                         transition.getTarget().getId() == States.PROVINCE) {
56                     logger.info("省份:");
57                     return;
58                 }
```

```
59
60
61                  if (transition.getSource().getId() == States.START &&
62                          transition.getTarget().getId() == States.CITY) {
63                      logger.info("市:");
64                      return;
65                  }
66
67                  if (transition.getSource().getId() == States.PROVINCE &&
68                          transition.getTarget().getId() == States.CITY) {
69                      logger.info("市:");
70                      return;
71                  }
72
73                  if (transition.getSource().getId() == States.PRC
74                          transition.getTarget().getId() == States.DISTRIC) {
75                      logger.info("县:");
76                      return;
77                  }
78
79                  if (transition.getSource().getId() == States.CITY &&
80                          transition.getTarget().getId() == States.STREET) {
81                      logger.info("街道:");
82                      return;
83                  }
84
85                  if (transition.getSource().getId() == States.DISTRIC &&
86                          transition.getTarget().getId() == States.STREET) {
87                      logger.info("街道:");
88                      return;
89                  }
90
91                  if (transition.getSource().getId() == States.STREET &&
92                          transition.getTarget().getId() == States.STREET_NUM) {
93                      logger.info("街道号:");
94                      return;
95                  }
96              }
97          };
98      }
99
100     @Override
101     public void configure(StateMachineConfigurationConfigurer<States, Events> c
102         config.withConfiguration().listener(listener());
103
104     }
105 }
```

运行：

```
1  import org.springframework.beans.factory.annotation.Autowired;
2  import org.springframework.boot.CommandLineRunner;
3  import org.springframework.boot.SpringApplication;
4  import org.springframework.boot.autoconfigure.SpringBootApplication;
5  import org.springframework.statemachine.StateMachine;
```

```
@SpringBootApplication
public class Application implements CommandLineRunner {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

    @Autowired
    private StateMachine<States, Events> stateMachine;

    @Override
    public void run(String... args) throws Exception {
        stateMachine.start();
        stateMachine.sendEvent(Events.HAS_PROVINCE);
        stateMachine.sendEvent(Events.HAS_CITY);
        stateMachine.sendEvent(Events.HAS_STREET);
        stateMachine.sendEvent(Events.HAS_STREET_NUM);
    }
}
```

结果:

```

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.5.6.RELEASE)

2017-09-23 17:55:27.787  INFO 9344 --- [           main] xtipc.statemachine.adc
2017-09-23 17:55:27.790  INFO 9344 --- [           main] xtipc.statemachine.adc
2017-09-23 17:55:27.864  INFO 9344 --- [           main] s.c.a.AnnotationConfig
2017-09-23 17:55:28.565  INFO 9344 --- [           main] trationDelegate$BeanPo
2017-09-23 17:55:29.075  INFO 9344 --- [           main] o.s.j.e.a.AnnotationMB
2017-09-23 17:55:29.079  INFO 9344 --- [           main] o.s.c.support.DefaultL
2017-09-23 17:55:29.089  INFO 9344 --- [           main] xtipc.statemachine.adc
2017-09-23 17:55:29.092  INFO 9344 --- [           main] o.s.s.support.Lifecycl
2017-09-23 17:55:29.092  INFO 9344 --- [           main] o.s.s.support.Lifecycl
2017-09-23 17:55:29.097  INFO 9344 --- [           main] xtipc.statemachine.adc
2017-09-23 17:55:29.098  INFO 9344 --- [           main] xtipc.statemachine.adc
2017-09-23 17:55:29.098  INFO 9344 --- [           main] xtipc.statemachine.adc
2017-09-23 17:55:29.098  INFO 9344 --- [           main] xtipc.statemachine.adc
2017-09-23 17:55:29.099  INFO 9344 --- [           main] xtipc.statemachine.adc
2017-09-23 17:55:29.100  INFO 9344 --- [       Thread-2] s.c.a.AnnotationConfig
2017-09-23 17:55:29.101  INFO 9344 --- [       Thread-2] o.s.c.support.DefaultL
2017-09-23 17:55:29.102  INFO 9344 --- [       Thread-2] o.s.s.support.Lifecycl
2017-09-23 17:55:29.102  INFO 9344 --- [       Thread-2] o.s.s.support.Lifecycl
2017-09-23 17:55:29.103  INFO 9344 --- [       Thread-2] o.s.j.e.a.AnnotationMB
2017-09-23 17:55:29.112  INFO 9344 --- [       Thread-2] o.s.s.support.Lifecycl
```

顶 1 踩 0


- 上一篇 Elasticsearch 集群优化总结
- 下一篇 Spark机器学习环境搭建



相关文章推荐

- 基于Spring-statemachine的有限状态机(FSM)的介绍…
- MySQL在微信支付下的高可用运营--莫晓东
- 使用Spring StateMachine框架实现状态机
- 容器技术在58同城的实践--姚远
- 使用Spring StateMachine框架实现状态机
- SDCC 2017之容器技术实战线上峰会
- 使用Spring StateMachine框架实现状态机
- SDCC 2017之数据库技术实战线上峰会
- 数学之美系列十：有限状态机和地
- 腾讯云容器服务架构实现介绍--董
- Android蓝牙源码分析——StateMachine状态机
- 微博热点事件背后的数据库运维心得
- quick-cocos2d-x游戏开发——StateMa
- StateMachine状态机
- Mina状态机介绍(Introduction to mina-statemachine)
- quick-cocos2d-x游戏开发——StateMachine状态机


查看评论

暂无评论